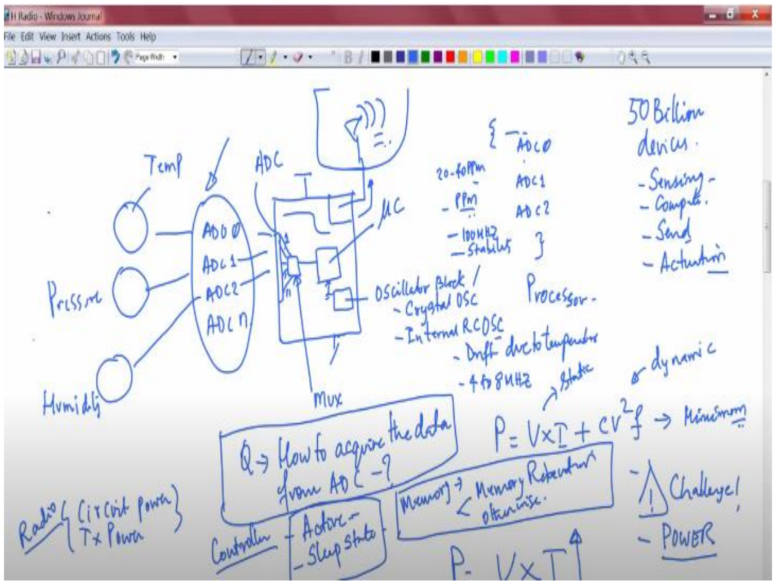
Design for Internet of Things

Prof. T. V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science, Bengaluru

Lecture - 04
Challenges Part- 02

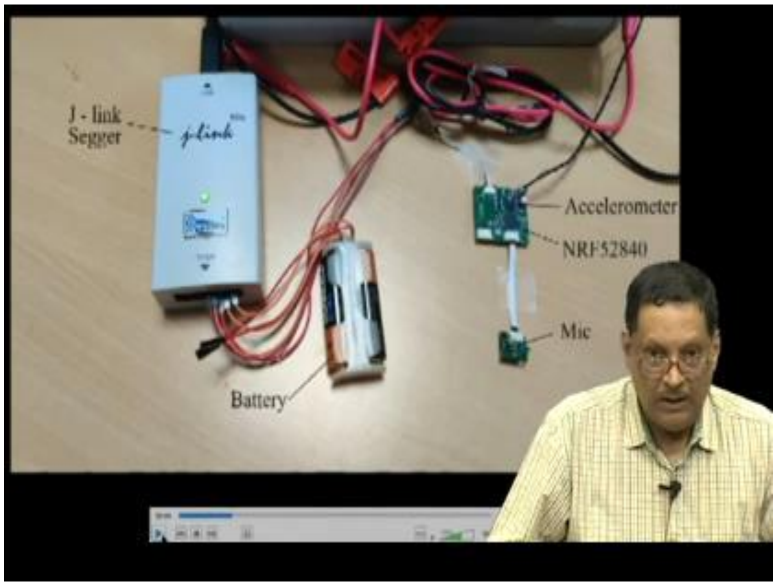
While we will answer this question related to how to acquire data from ADC, we have already shown this picture where there are several ADC sensors.

(Refer Slide Time: 00:38)


H Radio - Windows Journal
File Edit View Insert Actions Tools Help
Temp
Pressure
Humidity
ADC 0
ADC 1
ADC 2
ADC n
ADC
μC
Mux
{ ADC 0
ADC 1
ADC 2 }
20-40PPm
- PPm
- 100MHz
- Stabilis
Oscillator Block /
- Crystal OSC
- Internal RC OSC
- Drift due to temperature
- 4 to 8MHz
Processor -
50 Billion devices.
- Sensing -
- Compute.
- Send
- Actuation
static
dynamic
$P = V \times I + CV^2 f$ → Minimum
Q → How to acquire the data from ADC -?
Memory → < Memory Retention / otherwise.
Radio { Circuit power / Tx Power }
Controller - Active - / - Sleep State
$P = V \times I$
Challenge!
- POWER

So, you have ADC 0 to ADC n there are multiple ADC ports to which the sensors are connected. It may be useful to just see a quick demo and appreciate what exactly all this whole thing is actually playing out. So let me go to the demo.

(Refer Slide Time: 01:02)



Look at what is here right now. So, we have the programming part which is essentially called J-link Segger. This is the debugger which essentially will connect to a PC, and you can write all the embedded C code for your application and then download using the J-link Segger onto the target board. The target board is on the right side, what you see here, you see that there are two sensors here. One is an accelerometer which is perhaps connected to the digital port of the system.

And a microphone sensor which you see down below is actually one of the analog sensors. The microcontroller SoC of interest is nRF52840. I am not advocating any specific controllers in this course. But it is just that this is some controller that we have been using for a little while. So, I thought we should continue the lab exercises with this particular controller. And then there is a battery which is powering the small SoC board that you see here.

And so, the experiment is to just see how the current consumption actually varies as we go along in its operation. Now what is the operation of interest? The operation is to begin with this controller is in sleep. And when it is in sleep it consumes some amount of current. And then periodically it wakes up and then acquires the microphone data and takes it inside the microcontroller for further processing.

Periodically, it is either it is waking up and looking at the microphone signal and then acquiring that data. You can also configure it that microphone can be used only when microphone signal acquisition will happen only when it crosses a certain threshold in the sound level. That also you can do but this is a very simple demo. In fact, putting up a threshold-based detection is perhaps the right thing to do so that you will avoid a lot of noise. And you would not be acquiring data when it is quiet.

If you do periodic acquisition even the quiet environments have to be picked up and then you have to process it which may be a little bit overhead in terms of battery power. So, by setting a certain threshold, adjusting the threshold such that it is over and above the noise only then you say that it is a signal and then acquire that signal may be the right thing to do. But let us do it step by step. We will take this very simple example. Now let me play it.
(Video Starts: 04:06)

You will see as we go along there is a certain indication of the current consumption that is showing up. You can see now the value there is shown as 0.39 or about 0.4 milli amperes of current. It appears to be the controller appears to be in some sleep mode, some sleep mode. I will be very careful here. It is in some sleep mode. And therefore, we will not elaborate anything further than that. It is in some sleep mode. Why we say it is in sleep mode?

You will realize that it is in sleep mode because the system will show higher current. Now you can see if I go a little back and play it, so we missed it. So, you will see that it has indicated you can see this. This value is 1.48 milli amperes. It has taken more than 1 milliampere from what we had seen earlier which was only 0.4 milliamperes. So therefore, this is in some other mode in the microcontroller. It is in some other mode. It is in what mode?

Definitely the ADC block is enabled, and it is acquiring the ADC data from the system that definitely is what is happening. It is still not communicating because we did not do any further processing. We are just taking the data from the ADC port into the controller and that is what is actually being depicted here. And then after it acquires the data it perhaps goes into another sleep mode. And this time I show you a snapshot of not the 0.4 but the 0.2 milli amperes.

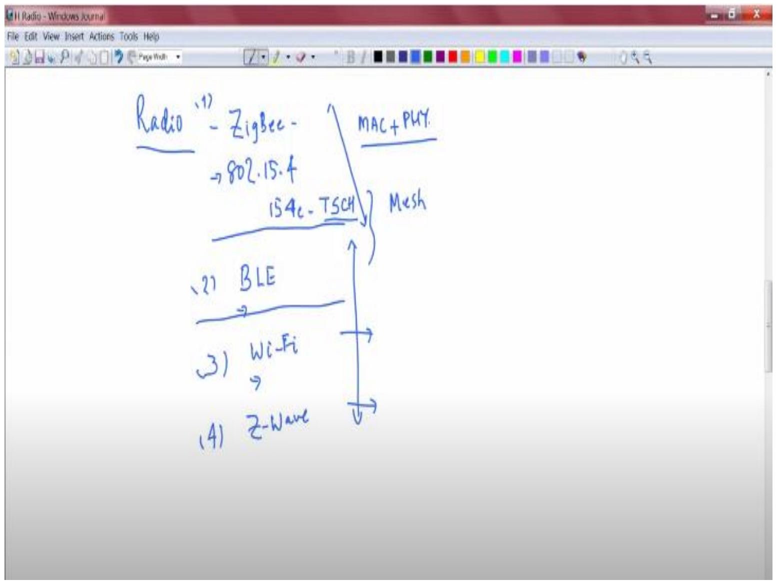
That means it must be going to even deeper sleep mode from what we started with. So now this is what will go on in a cycle because a controller has been told to do something. It repeats this whole process, goes and acquires data and goes into one sleep mode where it consumes 0.4 milli amperes and then acquires data, then it goes back and goes into another deep sleep mode where it goes to 0.2 milli amperes and so on.

(Video Ends: 06:19)

So, this demo which we have seen essentially looked at the microphone and it was trying to acquire the data. And we did not show anything with respect to communication. See in other words, it only acquired the data. But this radio part, we never looked at the radio part, we only looked at the acquisition part. Still, we have not answered this question. How to acquire the data from ADC? This is a very deep question by the way which we have to be very careful.

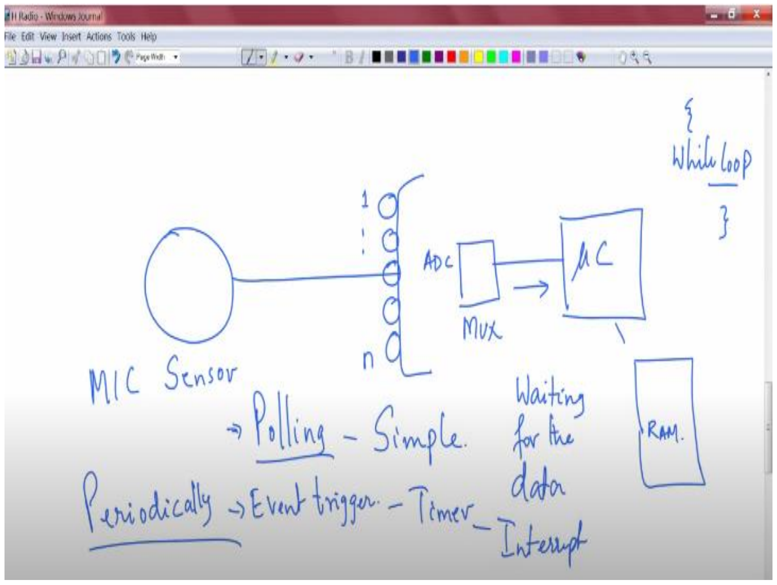
The demo is not illustrating anything. The demo is simply saying the reader acquired the data. How it has acquired the data is not clear. So let us go into some detailing there.

(Refer Slide Time: 07:08)



And then we will come to the other parts as we go along.

(Refer Slide Time: 07:16)



So let me take a fresh page and explain to you how this ADC system this is your sensor. I am just expanding this, and this is the ADC. There are many ports. And we have connected the particular sensor to one of the I will put n here to one of them. We mentioned that there is a MUX here, a multiplexer and this is going to the microcontroller. Now think of the following folks. You should acquire this data and store it in the RAM.

You are to acquire the data and put it in RAM. One option is you write a code here which says periodically you run the code such that you acquire the data from the microphone and then you are giving a command, it will acquire the data, the data comes and then sits here. This is one way of doing it. This way of acquiring the data by writing a code to go to the microphone sensor and acquire the data is actually called polling. This you can do polling.

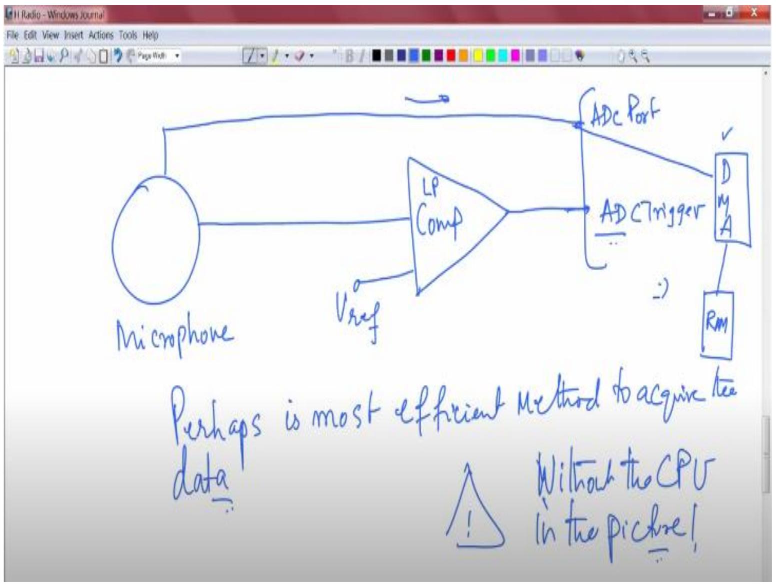
What happens during the acquisition time? The command is given by the controller to the ADC to acquire the sensor values and during that time the controller is on, waiting for the data to come back. It is doing nothing. It is just waiting for the data to come back. It can be in orders of hundreds of microseconds and that amount of time the controller is ON waiting for the data. Here is the problem. It is waiting for the data, and it is not doing any computation which means if you go back to this equation this is where the trouble is.

This P which we wrote V cross I which is the static power. This is I. This static power of the processor starts to increase because it is doing nothing. No computation is happening, and nothing is actually just waiting for the system to respond. So, this static power will definitely go up. And it will increase the power requirement for the control for the complete system and thereby reduce the battery life of the embedded IoT node that you are looking at.

So that is a problem. But it is very simple in terms of code. It is very simple. All you have to do is put a while loop, acquire the data and you are done. So, I will put here a while loop. Correct? All you need is a while loop, very simple. Another way of acquiring this data is event triggering. Event triggering means you put a timer and let the timer generate an interrupt. Let the timer generate an interrupt and once the timer interrupt is generated you go and acquire the data.

You can do that. This is typically used whenever you want to acquire the data periodically. Every time you need to acquire data periodically you put a timer and you trigger the timer, the controller will go into the interrupt handling mode, there a small program is written to acquire the data, it acquires the data and puts it to RAM and then exits the interrupt routine and goes back to normal working. That is one way of doing it. Another way is to do it as I mentioned, particularly for microphone sensor to put a threshold. Now I will draw it again.

(Refer Slide Time: 12:40)


ADC Port
D M A
LP Comp
ADC Trigger
RAM
Microphone
Vref
Perhaps is most efficient Method to acquire the data
Without the CPU in the picture!

This is the microphone and this microphone signal you put it to what is known as comparator. Put it to a comparator. These comparators are typically low power comparators. Extremely low power comparator. The output of the comparator you will connect this to the ADC, trigger input to the ADC and the data directly is connected to the particular ADC port. Data from this ADC port is acquired only when this ADC trigger comes.

Not otherwise. Trigger should come and trigger will come only when comparator triggers and comparator itself will be set with the threshold. In electrical terms this is actually called you give it what is known as a reference. You apply some reference, and you compare it with that reference. If it is greater than that you get a signal trigger here and then you acquire the data. This perhaps is most efficient to acquire the data.

You may now have one critical doubt in your mind. The critical doubt will be in all these discussions the CPU is still there. CPU has to take the effort of taking the data from the ADC port and then storing it to RAM. That problem is still there that means static power can continue to be an issue. Even if you do all these things that little period over which the data is being acquired by the ADC CPU has to intervene and has to store the data into RAM. What is the savings we will get? That is a question. Actually, I only gave you the half-truth.

The actual truth is you do not need to get the CPU into the picture at all to do this. And the way you do it is you get rid of the CPU block and replace it with the DMA block. And DMA will take the trouble of writing to RAM. I will remove this line and I will put it here. Now it is the business of DMA which is another ultra-low power block within the controller to actually take the load off checking this output, getting the ADC to trigger, acquiring the data and storing it into RAM without the; what is important CPU in the picture.

This is important. And this gives you a tremendous amount of power savings. And folks, simple things systems which are already there on controllers have the datasheet you have to read carefully, understand the different blocks which are there inside the controller, put them to use as effectively as possible so that you will be able to configure your embedded system for an extremely high lifetime.

Because you are doing all these interventions. Recall now in this particular picture CPU can be doing something else because DMA is the one that is actually ensuring that the data is stored in RAM. Or CPU could be in a deep sleep state and DMA is doing the business all by itself. And thereby you are again saving energy. Either you are putting the CPU to deep sleep and let the DMA do it or CPU can be doing something else which is also an important thing.

You have to make the CPU be very efficient and it is working. You cannot block it and get it to do only a specific activity. So therefore, this is what will actually constitute tuning systems for ultra-low power. And we will go into details of these systems. And I will explain to you several of the settings which you can do. We will go into a little more detail about some data sheets particularly with respect to the Nordic controller systems.

You know this picture. We drew this picture about a sensor and then corresponding input to a low power comparator which is in turn going to an ADC. The comparator output actually triggers the ADC and then data is acquired through this channel. Analog data is acquired through this channel and then DMA is initiated and return to RAM and then the CPU is actually interrupted.

That clearly indicated that this is perhaps the most efficient method to acquire data without the CPU in the picture. All this acquisition of data we mentioned is without the CPU in the picture which is a very strong point of embedded controllers. You have to exploit these features. Now are we talking in the air or are there such processors with all these capabilities is the question. So, for that what we will do?

I will show you one specification of one SoC which is the Nordic 52840. Most of the blocks I have explained here in this picture are actually from that particular Nordic ADC which is 52840.
(Video Starts: 20:12)

So let us see what are all the features. Several of the features that we described in the earlier class actually are showing up here. You can see the Bluetooth specification which is the wireless connectivity which I had shown here if you look at this screen, this screen essentially is the one that is talking about the wireless connectivity. So, this screen is here. So, this is the wireless connectivity part.

And if you go back to this screen now you will see that it is showing the Bluetooth 15.4-2006 version of the standard. Yes, then you have all the related parameters related to the radio here. And also, the radio chip is configurable to support not just Bluetooth 5.0, but it can also support IEEE 802.15.4-2006 version of the standard which is popularly people call it also the ZigBee standard. So, this radio here is configurable to either this or that. Then there is the controller.

If you look back at this picture again this screen that we have here, you see, we mentioned about the controller, this is the controller, and this is the ADC we just described the radio part. If you look at the controller and come back to this screen, this screen is talking about it is a Cortex-M 32 processor with the floating-point unit built inside, 64 MHz CPU and it has a certain amount of performance.

It is called 212 EEMBC CoreMark performance. We will come to that as we go along. And now another important parameter which you must eyeball in is these 52 micro amperes per megahertz. This is actually the crux of the whole power consumption as far as the CPU is concerned. It is talking for the kind of performance parameter which there are multiple ways by which you benchmark a CPU.

CoreMark is one way of benchmarking a CPU. At this particular SOC and the CoreMark if you are running CoreMark from flash memory the processor consumes 52 micro amperes for every one megahertz of operation. You could easily multiply it into 64 in order to get the overall current consumption when it is running at full frequency. So that is essentially what you want to interpret it as.

Then there are other timers there is a watch point and the trace debug modules and so on and so forth. Then there is a rich set of security features. There is the Arm TrustZone which is quite well known, and it seems to support the Arm TrustZone as well. Then let us come to other points here.

And here if you will go up you will see flexible power management. This controller can operate from 1.7 to 5.5-volt supply voltage range.

In other words, if you have two 1.5-volt cells which gives you 1.5 each. If you have two 1.5-volt cells from 3 volts onwards it can work and it can also work if the 3 volts goes down to as low as 1.7. So, think about some examples like two batteries of 1.5 each discharging continuously and coming down to 1.7 which is slightly more than a single cell. Single cell gives you only 1.5. So, it can work up to 1.7. No problem.

And it can work up to a maximum of 5.5 volts supply range. Now each one of these statements here are loaded. Look at this statement. It says on-chip DC-DC and LDO regulators with automated low current modes. That means it can give you, you can trade performance to noise, you can trade performance to energy here. So, this is the most important thing. See if you take on-chip DC-DC the whole processor will consume less current as compared to LDOs.

LDOs are low dropout regulators. These are linear regulators, and they give you at a maximum of 50% efficiency whereas DC-DC converters are switching regulators and they can give you 95 to 97% efficiency. So on-chip efficiency I am talking of power efficiency. So that is a very important point. Power efficiency can be as high as 95% and greater than 95%. So if you use DC-DC you will get a long lifetime as compared to using LDOs.

What is the problem of using DC-DC? Why is there an LDO option? These are switching regulators and therefore they introduce a lot of switching noise and that can become a non-starter particularly if you have non-DC kind of analog inputs which we will come to because there are demonstrations which will show you that you are going to interface analog components to it. So, in that situation if the analog voltage generated is in the order of a few millivolts.

And so on and if the noise emanating from DC-DC buck converters are also in that range, then you would not get any useful signal. Everything converted is just noise out there. In such situations one would use an LDO and would not use a DC-DC converter. But if you choose an

LDO, the power consumption goes up. The power efficiency is low. So, this design for the IoT course will actually be talking about these tradeoffs.

When should you choose LDO? I should choose an LDO under certain conditions. I will choose this LDO if I know the LDO very well. It is working very well. I will choose the DC-DC converter buck converter if I know it is functioning and its operation very well. So, it is important for us to know these operations very well. So, coming back it says 1.8 volts to 3.3 regulated supply for external components.

That means it can also give you a Vout which essentially you can drive several components as well. So, it can source these voltages as well. For example, a controller has to be interfaced to let us say a display system. Then where do you generate that output voltage? You can directly take it from this controller itself because it can give you this output voltage as well. And then there are other features and look at so many parameters are all energy related.

It says 0.4 micro amperes at 3 volts in System OFF mode, no RAM retention. 1.5 micro amperes at 3 volts in System ON mode, no RAM retention, wake on RTC. So current consumption options are made available to us. And we need to exploit these features as much as possible. The difference between the 0.4 micro amperes at 3 volt and 1.5 micro amperes is if the system is off the only way is to go and reboot that system and buy a hard reboot.

That is someone physically will have to go and press a button so that it will come back. And of course, there is no way by which the data that is required is there sitting in the RAM. Whereas with the 1.5 micro amperes at 3-volt operation system is on which means you can use software timers to trigger the system to wake up automatically and maybe want to acquire some data at regular intervals and so on. So, at that instant you can trigger it with this.

So, you do not need human interventions in order to come to this kind of extremely low current consumption. The point of discussion, you can go on reading the specifications, but the point of discussion is not so much at least at the moment is not so much about the other things. But it is mostly to do with the ADC. This is the key crux of the whole story. Because we are attempting to connect a sensor to our ADC and therefore it is important to come from a data sheet perspective on the ADC that is to be used for our work.

But before you actually go into the details of the ADC also do not forget that if you go back to these pictures, this picture is talking about ADC. This particularly this point here where a sensor this is your sensor. The sensor is being acquired through ADC. It is tightly coupled to another component called the DMA. And why do you need this DMA? The processor offloads its activity related to acquisition of data from this microphone to this RAM without the CPU.

So, a DMA can do the job. Direct memory access, it supports multiple channels. And therefore, whenever we talk about ADC you will talk about DMA because together, they can achieve what is known as the ultra-low power operation for your systems. Therefore, you must know about the ability of the system to support the DMA as well. So ultra-low power. So let us shift back. You will see that this Nordic controller actually supports DMA and where is it written?

Here you are. Easy DMA automated data transfer between memory and peripherals. That is the crux. That is why the other picture was very useful for you. So, you can say that an ADC is a peripheral connected to it and easy DMA can be configured to acquire data from the ADC. Now we have to go and look at why so much noise we are doing about this ADC. The simple reason is you have to understand its ability to acquire data on these embedded controllers.

Because IoT devices actually will have sensors connected and therefore understanding the ADC becomes absolutely critical. Look at what the datasheet says. It says it will offer you 12-bit resolution, 200 kilo samples per second. There are 8 configurable channels with the programmable gain. So, these are all I mean you have to understand every part of it. Only then you can actually get to understanding how to use this ADC as effectively as possible.

So, there you are. So, we will come to that and understand these things. And you can also look up other specifications of the controller. So now let us move our focus to understanding a little bit more about ADCs. So, in the next instance I will be showing you something a little more deeper into ADCs. Thank you very much.

(Video Ends: 32:28)